

# Energy-Efficient Scheduling: Classification, Bounds, and Algorithms\*

Pragati Agrawal  
pragati.a.in@ieee.org

Shrisha Rao  
shrao@ieee.org

## Abstract

The problem of attaining energy efficiency in distributed systems is of importance, but a general, non-domain-specific theory of energy-minimal scheduling is far from developed. In this paper, we classify the problems of energy-minimal scheduling and present theoretical foundations of the same. We derive results concerning energy-minimal scheduling of independent jobs in a distributed system with functionally similar machines with different working and idle power ratings. The machines considered in our system can have identical as well as different speeds. If the jobs can be divided into arbitrary parts, we show that the minimum-energy schedule can be generated in linear time and give exact scheduling algorithms. For the cases where jobs are non-divisible, we prove that the scheduling problems are NP-hard and also give approximation algorithms for the same along with their bounds.

**Keywords:** scheduling, energy efficiency, distributed systems, computational complexity

## 1 Introduction

Energy is a precious resource of the current industrial economy. For energy conservation and sustainability, it is necessary to achieve energy efficiency, which we may take to mean minimization of the energy consumed to do a given amount of work. This requires ways to achieve greater energy efficiencies by proper scheduling of jobs over machines, in addition to making each machine individually more energy efficient. Scheduling theory is an important field of study that has received attention from researchers for decades, but most work on scheduling till now has only looked at makespan and other time-related objectives. Energy being such a limited resource and directly related to cost, it can in certain instances be considered more important than time, i.e., it may be better to complete jobs using minimum energy, rather than completing them quickly. Clearly, scheduling jobs in such a way that

---

\*A part of this work is presented in a workshop paper (Agrawal and Rao, 2015) at the 4th International Workshop on Energy-Efficient Data Centres (E2DC 2015), co-located with ACM E-Energy 2015, Bangalore, India.

the total energy consumption of the system is minimal, or at least low, is of the essence. We present a generalized theory for energy-efficient offline scheduling in this paper, which does not go into domain-specific issues or technologies.

Our work is applicable to systems whose machines are similar in their capabilities but can have different working and idle power consumptions. The speeds at which machines can execute tasks can be different from one another, but are not time-varying, i.e., each machine operates at the same speed whenever working. All machines are connected to one another, so that a job from one machine can be transferred to any other machine. There is no set limit to the number of machines in a system. Machines and jobs to be executed are independent, and hence any job can be executed in any order on any machine. Each machine is characterized by a fixed working power (the power consumed by it when under load) and a fixed idle power (the power consumed by it when running idle), as seen in previous work (Agrawal and Rao, 2014). Given a set of interconnected machines and independent jobs, our results specify the allocation of jobs on the machines so that the total power consumption of the system is minimized. However, unlike that previous work, we focus on exact results rather than using heuristics for scheduling.

Considering a system of machines cooperatively running similar loads, we classify the problems of energy-minimal offline scheduling with non-identical interconnected machines and independent jobs as follows:

1. *Identical speeds, divisible loads:*

This class of problem is discussed in Section 5.1.1, where we derive results to prove that energy-minimal scheduling under this type of system can be solved in linear time; we also give an algorithm for the same.

2. *Identical speeds, non-divisible loads:*

This turns out to be NP-hard, as shown in Section 5.1.2. We also give a linear-time approximation algorithm which gives a solution within a bound of  $\frac{4}{3}$ .

3. *Different speeds, divisible loads:*

In Section 5.2.1 we give results for energy-minimal scheduling on such a system and also a linear-time algorithm to find the minimum-energy schedule.

4. *Different speeds, non-divisible loads:*

As with the case of identical speed machines, this class of problems is also NP-hard. We give a linear-time approximation algorithm for this class in Section 5.2.2 which gives a result within the bound of  $\frac{19}{12}$  times the optimal.

We first introduce a generic system model which abstracts the relevant aspects of energy-efficient scheduling. We then give a classification of the problems of energy-efficient scheduling on different types of systems, and study their complexities. This makes possible analyses giving results which govern the relative distribution of work among machines for

maximum energy efficiency, given the energy specifications of the machines. Using these results we design scheduling algorithms for different classes of systems. For problem classes in which scheduling is NP-hard, we give approximation algorithms along with their bounds.

We deliberately do not give specific units for energy (which is typically measured in joules, ergs, kilowatt-hours, etc.), power (often measured in watts, etc.), work (generally specified in the same units as energy), or speed (work done per unit time); we likewise do not find it necessary to be precise about types of energy or energy sources. This is so to remain agnostic towards particular system technologies and instantiations, and to avoid domain-specific biases in our analyses.

There is a lot of work done for makespan scheduling, and other time-related objectives. But solving for time-related objectives does not guarantee minimum energy. The novelty of this work is in considering energy specifically with completely generic machine specifications. No other theoretical work until now has considered energy consumption of machines while idle, which is known to be significant from real life; e.g., in data centers (Hirofuchi et al., 2010; Berral et al., 2010). Our system model also shows the absolute bounds of energy-minimal scheduling for specific types of systems. We derive results (for divisible jobs) which serve as an upper-bound to maximum achievable efficiency—because a system with divisible jobs can always have higher efficiency as compared to system with non-divisible jobs (Pinedo, 2012).

A part of this theoretical framework, seen here in Section 4.1, has been presented in our workshop paper (Agrawal and Rao, 2015), which introduces the classification of energy-minimal scheduling problems and gives results on the class of systems which have identical speed machines and divisible loads. It also gives results for a special case of system with different speed machines and divisible loads in which idle power consumption of machines was considered to be zero. It can be seen as the beginning of the work presented in the current paper which comprehensively covers most types of energy-minimal scheduling problems with completely interconnected machines and independent jobs.

In the present paper we deal with theoretical aspects of energy-minimal scheduling, deriving results which can guide the design of scheduling algorithms for systems. On any given schedule, if the conditions indicated in the results are satisfied, then that schedule will be energy-minimal. Likewise, the results can also indicate when a system of machines would be inherently wasteful of energy, and can thus guide more energy-efficient system designs.

It is easy to see (even a back-of-the-envelope calculation can suffice) that there are many cases where it is not appropriate to run all machines, if one’s goal is to complete a set of jobs with minimal energy consumption. The question that arises is: given a set of jobs to complete, which machines should one use, and which not? Our results and analyses address this point.

Machines not in use are not assumed to be switched off, but do consume power (what we call idle power). We have deliberately allowed that machines do not get switched off when not in use, given the reality of machines in many domains (after all, the on-off cycle

time for an industrial furnace, or even a personal computer, is not small). The situation where a machine does get switched off is merely a special case where the idle power of that machine is zero.

For the class of system in which machines have identical speeds with different power ratings, and when the jobs are divisible, we first determine the precedence of the machines (the order in which they should be assigned work). We then derive the optimal number of machines which should be working so that the energy consumption is minimum. Based on this we derive what should be the optimal allocation of work among the machines chosen to run. We give an  $\mathcal{O}(m)$  (where,  $m$  is the number of machines) algorithm which instantiates the theoretical results, to derive the energy-minimal schedule given the specifications of the machines and jobs.

The slightly harder problem of finding the minimum-energy schedule for divisible jobs over machines with different speeds and power ratings can also be solved in  $\mathcal{O}(m)$  time. We give an exact algorithm for this class too.

For the classes of problems with non-divisible jobs, we see that the problems are NP-hard. Hence we give  $\mathcal{O}(m)$  approximation algorithms for both cases: systems with identical- and different-speed machines.

The remainder of the paper is organized as follows. Related work is mentioned in Section 2. The system model along with the notation used in defining the energy-minimal scheduling problem and the motivation is presented in Section 3. Section 4 discusses about the precedence of machines for both identical speed and different speed machine case. The proposed algorithms along with their bounds are given in Section 5. Finally, Section 6 describes the two types of measures commonly used for energy efficiency in systems, and shows that these are incompatible.

## 2 Related Work

In energy-minimal scheduling, as opposed to classical makespan scheduling, there are additional parameters (energy specifications) which come into picture and hence it is inherently more complex than makespan scheduling. Agrawal and Rao (2014) show that energy-efficient scheduling is strictly harder than makespan scheduling, and present a generic model for scheduling which takes into consideration both the working power and the idle power of each machine. They propose three heuristic algorithms for energy-aware scheduling in a system where jobs have precedence constraints.

Though there have been a lot of work done in the field of scheduling, the prime focus of general works on scheduling (Pinedo, 2012, 2009; Herrmann, 2006) has been to optimize on objectives related to time—such as makespan, earliness, and avoidance of tardiness.

Existing literature dealing with scheduling for energy reduction generally focuses on specific domains, such as communication networks, embedded systems, and high-performance computing. These works invariably rely on specific features and technologies of those

domains. For instance, Wang and Saksena (1999) address parallel task scheduling for reducing energy consumption for high-end computing using the DVFS (dynamic voltage and frequency scheduling) technique with some heuristics. More recently, Yassa et al. (2013) also base their approach on DVFS to minimize energy consumption (as do many others). Huang et al. (2011) propose energy-aware task allocation using simulated annealing with timing adjustment for network-on-chip based heterogeneous multiprocessor systems. Sheikh et al. (2012) survey existing scheduling algorithms and software for reducing energy dissipation in performing tasks on different platforms including single processors, multicore processors, and distributed systems. This survey is quite comprehensive and discusses the architectural, software, and algorithmic issues for energy-aware scheduling of computing devices. Pouwelse et al. (2001) give a heuristic for energy priority scheduling for variable voltage processors. Bambagini et al. (2013) use the combination of offline-DVFS and online-DPM (dynamic power management) techniques to reduce energy consumption in embedded systems. Some recent anthologies (Zomaya and Lee, 2012; Pierson, 2015) deal with energy efficiency in computing systems at various scales (processors to data centers). However, while they are rich in pragmatic approaches and domain-specific ideas for energy efficiency, they do not present a theory of energy-efficient scheduling *per se*, and only basic algorithms (e.g., Greedy-Min, Greedy-Max) are considered on occasion. It is patently obvious that even though there are works on application-specific energy-efficient scheduling, a generalized theory is yet far from developed.

In the domain of microprocessor power management and energy saving, much of the prior work uses speed scaling and power-down techniques (Irani and Pruhs, 2005; Albers, 2009). Augustine et al. (2004) suggests optimal power-down strategies with more than one low-power state. When a device is idle, their algorithm turns the device in low-power sleep state. Bansal et al. (2009, 2013) use the other technique called speed scaling. Bansal et al. (2007) have given a tighter bound for the YDS algorithm given by Yao et al. (1995) and proved that their BKP algorithm is cooling-oblivious. Bansal et al. (2009, 2013) consider an arbitrary power function of speed, for speed scaling.

There are some studies of scheduling whose theoretical developments can help in energy-efficient scheduling too. One such is divisible load theory, which studies methodologies involving the continuous and linear modeling of partition-able communication and computation loads for parallel processing (Veeravalli et al., 2003; Yu and Robertazzi, 2003; Singh and Sahu, 2014; Abdullah and Othman, 2013; Robertazzi et al., 2005). Another such field is multi-objective scheduling, in which machine cost is considered as one of the objectives. (In general, machine cost includes any cost which incurs because of usage of the machines of the system.)

There is some prior work concerning scheduling for multiple criteria. For instance, Leung et al. (2012) consider two objectives for scheduling: one is time-related such as makespan (as the customer’s objective), and the other is total machine cost (as the service provider’s objective). They form different final objectives by combining these two objectives in various ways and analyze the complexity of the scheduling problem for each of these

various combinations. Lee et al. (2014) propose a heuristic for bi-criteria scheduling with machine assignment costs with worst-case performance bounds. Lee et al. (2012) also work on multi-objective scheduling. Other than makespan, they also minimize the total congestion and completion time, the maximum and total tardiness, and the number of tardy jobs. They give a coordination mechanisms for parallel machine scheduling. Shi et al. (2012) formulate an assignment scheme for the divisible load theory for sensing workload allocations. Drozdowski et al. (2014) give a method of visualizing the relationships between computation and energy consumption (in supercomputing and HPC applications) as two-dimensional maps similar to isotherms.

Though the objectives considered in these and other such works can be seen as marginally relevant to minimizing energy (setting energy consumed by a running machine as a machine cost), they have not considered the idle power consumptions of machines, which is needed for realistic analyses. In many papers that consider multiple scheduling objectives, all objectives are concerned with time only, and do not make sense for energy-minimal scheduling. The theoretical framework and classification of energy-minimal scheduling problems is not discussed in or derivable from any of these papers.

Clearly then, there is a need of a theoretical framework, as presented here, which realistically models systems of machines considering both idle power and running power, and also allows us to classify the various types of problems involved in energy-minimal scheduling and understand their complexities. This theory can of course also be extended for further fine-grained analyses (e.g., for machines with multiple performance levels). The theoretical framework proposed is thus a necessary addition to existing and ongoing work focusing on heuristic approaches to energy-efficient scheduling, and also complements technology-specific approaches to energy reduction such as DVFS (Le Sueur and Heiser, 2010).

### 3 System Model

In this section we formally introduce our system model and define the energy-minimal scheduling problem in the most general sense. As any other scheduling problem, we have a set of interconnected machines and a set of jobs that are to be executed using the machines. We presume that all machines are connected in such a way that a job can be transferred from any machine to any other machine without any energy dissipation. Jobs are independent and hence can be executed in any order and on any machine.

We consider two types of jobs, divisible and non-divisible. There can be two types of systems as well, based on the speeds of the machines in the system: one in which all machines run at the same speed (even though their power ratings may be different), and the other class in which machines run at different speeds. Thus, based on the type of job and the type of system, we have four classes of scheduling problems.

Let the set of  $m$  machines in a system be denoted by  $\mathcal{C}$ , where  $\mathcal{C} = \{c_i : 1 \leq i \leq m\}$  and  $c_i$  denotes the machine  $i$  of the set  $\mathcal{C}$ . We discuss the precedence of machines later in this

paper.  $\mathcal{R}$  denotes the working set of machines. The working power of machine  $c_i$  is denoted as  $\mu(c_i)$ , and the idle power as  $\gamma(c_i)$ . The sum of the idle power of all the machines is given by  $\sum_{i=1}^m \gamma(c_i) = \Gamma$ . The speed of machine  $c_i$  is denoted as  $v(c_i)$ . The speed (throughput of work per unit time) of a machine is fixed throughout its working tenure.

All machines in a system work in parallel, and the maximum working time of a system to execute a given set  $\mathcal{P}$  of  $n$  jobs  $p_1$  to  $p_n$  is  $T$ , which is the makespan of the system for that set of jobs. If machine  $c_i$  works only for time  $\tau(c_i)$ , then the idle time of that machine  $c_i$  is given by  $T - \tau(c_i) = \kappa(c_i)$ . The amount of work done by machine  $c_i$  is represented by  $w(c_i)$ , where

$$w(c_i) = \tau(c_i)v(c_i) \quad (3.1)$$

The weight (processing time) of job  $p_j$  is denoted by  $\psi(p_j)$  and the sum of the weights of all jobs is given by  $\sum_{j=1}^n \psi(p_j) = W$ . Also, the sum of work done by all machines is equal to the total work to be done, i.e.,  $\sum_{i=1}^m w(c_i) = W$ .

When  $v(c_i) = 1, \forall i, 1 \leq i \leq m$ , then we say that the machines are identical in their working capacities or speed, implying that they can execute and complete any job given to them in equal time. Also, in this case  $\tau(c_i) = w(c_i), \forall i, 1 \leq i \leq m$ . And if in the present case all jobs are executed sequentially on one machine, then that time taken is equal to  $W$ , the total amount of work to be done.

The energy consumed by a machine can be calculated given the power consumption by the machine in working state and the duration for which the machine works. Since the power rating of a machine is energy consumed per unit time, it is in principle possible to calculate the energy consumed by any machine over a given time duration by integrating the power consumption over time (Halliday et al., 2010). If the power consumed is constant over time, the energy is given by the product of power with time.

In many systems like data centers, it is not a simple matter to shut off machines, even when they are not given any load. Thus, if no machine is not switched off until the total work is over, even such an idle machine consumes some power though not producing any work. Considering the energy consumption in idle state too, the total energy consumed by a machine is the sum of the energy consumed in the working state and that consumed in the idle state.

The energy consumed by machine  $c_i$  in the working state is given by  $\mu(c_i)\tau(c_i)$ , and the energy consumed in the idle state by  $\gamma(c_i)(\kappa(c_i))$ . So the total energy consumption by a machine  $c_i$  is given by  $\mu(c_i)\tau(c_i) + \gamma(c_i)(\kappa(c_i))$ . With this, the expression for the energy consumed in a system can be given as the sum of the energies consumed by all  $m$  machines:

$$E = \sum_{i=1}^m [\mu(c_i)\tau(c_i) + \gamma(c_i)(\kappa(c_i))] \quad (3.2)$$

Symbol	Meaning
$E_{m,r}$	Total energy consumption of system with $m$ machines and jobs distributed to $r$ machines
$T$	Makespan of the system
$\mathcal{C}$	Set of machines $c_1, c_2, \dots, c_m$
$\mathcal{R}$	Working set of machines
$r$	Number of machines on which jobs are distributed
$\mu(c_i)$	Power consumption in working state by machine $c_i$
$\gamma(c_i)$	Power consumption in idle state by machine $c_i$
$v(c_i)$	Speed of machine $c_i$
$\tau(c_i)$	Time spent in working state by machine $c_i$
$\kappa(c_i)$	Time spent in idle state by machine $c_i$
$w(c_i)$	Amount of work given to machine $c_i$
$\mathcal{P}$	Set of jobs $p_1, p_2, \dots, p_n$
$\psi(p_j)$	Weight of job $p_j$
$W$	Total working time of all the jobs

Table 1: Notation

Putting the value of  $\kappa(c_i) = T - \tau(c_i)$ , in (3.2),

$$E = \sum_{i=1}^m [\mu(c_i)\tau(c_i) + \gamma(c_i)(T - \tau(c_i))] \quad (3.3)$$

The general energy equation can be formed by replacing  $\tau(c_i)$  with  $\frac{w(c_i)}{v(c_i)}$  (from (3.1)) in (3.3).

$$E = \sum_{i=1}^m \left[ \mu(c_i) \frac{w(c_i)}{v(c_i)} + \gamma(c_i) \left( T - \frac{w(c_i)}{v(c_i)} \right) \right]$$

After simplification,

$$E = \sum_{i=1}^m \left[ \frac{w(c_i)}{v(c_i)} (\mu(c_i) - \gamma(c_i)) + \gamma(c_i) T \right] \quad (3.4)$$

where  $1 \leq i \leq m$ .

The notation used is indicated in Table 1. (Though some terms are defined later in the paper, we list them here for quick reference.)

We assume the following in our system:

1. All jobs are independent, i.e., there are no precedence constraints between jobs. It means a job need not wait for completion of any to start its execution. Hence, there



is no idle time (gap) for a machine in between the execution of two jobs. This implies that

$$T = \max \tau(c_i), \forall i \quad (3.5)$$

2. In the class where jobs are divisible, the divisibility of jobs means, jobs can be arbitrarily divided and assigned to any machine.
3. All machines stay on for the duration of the makespan of the whole set of jobs; no machine is switched off while others continue to run. At any instant while the jobs are being executed, each machine  $c_i$  either consumes its working power  $\mu(c_i)$  if working, or its idle power  $\gamma(c_i)$  if not. (The situation where a machine gets switched off to consume zero power while not executing a job, can of course be trivially handled by setting  $\gamma(c_i)$  to 0.)
4. For the model in which the speed of all machines is the same, all machines are indexed in increasing order of the differences of their working power and idle power, i.e.,  $\mu(c_i) - \gamma(c_i) \leq \mu(c_j) - \gamma(c_j), \forall i, j$  where  $1 \leq i \leq j \leq m$ . For the model where machines can have different speeds, the machines are indexed in an order such that the first machine is the one with smallest  $\frac{\mu(c_i) - \gamma(c_i) + \Gamma}{v(c_i)}$  and afterwards machines are indexed in non-decreasing order of  $\frac{\mu(c_i) - \gamma(c_i)}{v(c_i)}$ , where  $1 \leq i \leq m$ .
5. For the model in which the speed of all machines is the same, one unit of work takes one unit of time for execution, irrespective of the machine on which it is executed.

The problem of energy-minimal scheduling refers to finding schedules for a set of jobs which would require minimum total energy when executed on a given system:

1. *Identical speed machines:*

The machines in the system have different working and idle power consumptions, but have equal speeds in execution of jobs. The aim for this type of system is to minimize  $E$  given by (3.3).

2. *Different speed machines:*

Machines can have different power consumptions as well as different speeds. The aim for this type of system is to minimize  $E$  given by (3.4).

For each of these classes we have two sub-classes, with divisible jobs and non-divisible jobs.

In any of these classes, the aim is to schedule a set of jobs on the given set of machines such that the energy consumption of the system  $E$  given by (3.3) and (3.4) is minimized.

The minimum makespan scheduling does not guarantee minimal energy schedules, and that in fact makespan scheduling is a special case of energy-minimal scheduling, where the

working power of machines is equal to their idle power. Putting  $\mu(c_i) = \gamma(c_i)$  in (3.3), we get,

$$\begin{aligned} E &= \sum_{i=1}^m [\mu(c_i)\tau(c_i) + \gamma(c_i)(T - \tau(c_i))] \\ &= \sum_{i=1}^m \mu(c_i)T \end{aligned}$$

This means that energy is directly proportional to makespan, and hence energy can be minimized by minimizing makespan if the working power is equal to idle power.

Consider an arguably more practical setting where idle power is less than the working power,  $\gamma(c_i) = z_i \cdot \mu(c_i)$ , where  $0 \leq z_i \leq 1$ :

$$\begin{aligned} E &= \sum_{i=1}^m [\mu(c_i)\tau(c_i) + \gamma(c_i)(T - \tau(c_i))] \\ &= \sum_{i=1}^m [\mu(c_i)\tau(c_i) + z_i\mu(c_i)(T - \tau(c_i))] \\ &= \sum_{i=1}^m \mu(c_i)[\tau(c_i) + z_iT - z_i\tau(c_i)] \end{aligned}$$

Then, the equation of energy can be written as:

$$E = \sum_{i=1}^m \mu(c_i)[z_iT + \tau(c_i)(1 - z_i)] \quad (3.6)$$

Hence, from (3.6) we can see that energy cannot be minimized by just minimizing makespan, though the energy consumption of an individual machine does depend upon its working time. Hence energy minimization is a more generic problem than makespan minimization.

In this section we have introduced the system model and the assumptions of the scheduling problem we have considered. We have also indicated the objective function which we will be looking to minimize. We now proceed to solve scheduling problems in the following section.

## 4 Precedence of Machines

In this section we first find the precedence of machines, i.e., the order in which they should be assigned work. This precedence helps in determining which machines should be allotted work and how much, to reduce the overall energy consumption of the system.

With differing power specifications of machines, it stands to reason that it may be advantageous to prefer some machines over others in doing jobs. We investigate the properties of the system to determine the precedence of machines so that the energy consumed in their execution is minimized. The problem of finding relative distribution of work among machines can be broken into two parts:

- Given the set of  $m$  machines, which subset of machines should be allowed to work and which should remain idle for all times during the makespan?
- What should be the distribution of work among the working machines?

There are no restrictions on the jobs except that the total load is considered to be  $W$ . This means that the results derived in this section are applicable to all type of systems. We first derive results for the case where machines have identical speeds and then for the case where they can have different speeds.

#### 4.1 Systems with Identical Speed Machines

In this subsection we consider the relative work distribution among machines with identical speeds such that energy consumption is minimized.

As indicated previously, we assume that all machines are indexed in the non-decreasing order of the differences between their working and idle power. We claim that when the machines are indexed in such an order, then the loading of machines should be such that the working times of these machines are in non-increasing order.

**Lemma 4.1.** *Given  $\mu(c_k) - \gamma(c_k) \leq \mu(c_l) - \gamma(c_l)$ , then for energy optimality of the system,  $\tau(c_k) \geq \tau(c_l) \forall k, l$ , where  $1 \leq k \leq l \leq m$ .*

*Proof.* We prove this by contradiction. Given  $\mu(c_k) - \gamma(c_k) \leq \mu(c_l) - \gamma(c_l)$ , then let us say that for minimal energy consumption,  $\tau(c_k) < \tau(c_l)$ .

With a little rearrangement of (3.3) we get,

$$E = \sum_{i=1}^m [(\mu(c_i) - \gamma(c_i))\tau(c_i) + \gamma(c_i)T]$$

This in turn gives:

$$\begin{aligned} E &= \sum_{i=1}^{k-1} [(\mu(c_i) - \gamma(c_i))\tau(c_i) + \gamma(c_i)T] + [(\mu(c_k) - \gamma(c_k))\tau(c_i) + \gamma(c_k)T] \\ &+ \sum_{i=k+1}^{l-1} [(\mu(c_i) - \gamma(c_i))\tau(c_i) + \gamma(c_i)T] + [(\mu(c_l) - \gamma(c_l))\tau(c_i) + \gamma(c_l)T] \\ &+ \sum_{i=l+1}^m [(\mu(c_i) - \gamma(c_i))\tau(c_i) + \gamma(c_i)T] \end{aligned} \quad (4.1)$$

The two possible cases can be: when  $\tau(c_k) \geq \tau(c_l)$ , and when  $\tau(c_k) < \tau(c_l)$ . We derive the energy equations for both cases and then compare them. By comparing the energy equations in both, we arrive at a condition under which the energy consumed in one case is less, and find a contradiction.

Case 1:  $\tau(c_k) \geq \tau(c_l)$ . Take  $\tau(c_k) = t + \epsilon_1$  and  $\tau(c_l) = t - \epsilon_1$ , where  $\epsilon_1 \geq 0$ .

Putting these values in (4.1) we get,

$$\begin{aligned}
E = & \sum_{i=1}^{k-1} [(\mu(c_i) - \gamma(c_i))\tau(c_i) + \gamma(c_i)T] + [(\mu(c_k) - \gamma(c_k))(t + \epsilon_1) \\
& + \gamma(c_k)T] + \sum_{i=k+1}^{l-1} [(\mu(c_i) - \gamma(c_i))\tau(c_i) + \gamma(c_i)T] + \sum_{i=l+1}^m [(\mu(c_i) \\
& - \gamma(c_i))\tau(c_i) + \gamma(c_i)T] + [(\mu(c_l) - \gamma(c_l))(t - \epsilon_1) + \gamma(c_l)T]
\end{aligned} \tag{4.2}$$

Case 2:  $\tau(c_k) < \tau(c_l)$ . Take  $\tau(c_k) = t - \epsilon_2$  and  $\tau(c_l) = t + \epsilon_2$ , where  $\epsilon_2 > 0$ .

Putting these values in (4.1) we get,

$$\begin{aligned}
E' = & \sum_{i=1}^{k-1} [(\mu(c_i) - \gamma(c_i))\tau(c_i) + \gamma(c_i)T] + [(\mu(c_k) - \gamma(c_k))(t - \epsilon_2) \\
& + \gamma(c_k)T] + \sum_{i=k+1}^{l-1} [(\mu(c_i) - \gamma(c_i))\tau(c_i) + \gamma(c_i)T] + [(\mu(c_l) \\
& - \gamma(c_l))(t + \epsilon_2) + \gamma(c_l)T] + \sum_{i=l+1}^m [(\mu(c_i) - \gamma(c_i))\tau(c_i) + \gamma(c_i)T]
\end{aligned} \tag{4.3}$$

Subtracting (4.3) from (4.2) to compare energies in both the cases, we get,

$$\begin{aligned}
E - E' = & [(\mu(c_k) - \gamma(c_k))(t + \epsilon_1) + \gamma(c_k)T] \\
& - [(\mu(c_k) - \gamma(c_k))(t - \epsilon_2) + \gamma(c_k)T] \\
& + [(\mu(c_l) - \gamma(c_l))(t - \epsilon_1) + \gamma(c_l)T] \\
& - [(\mu(c_l) - \gamma(c_l))(t + \epsilon_2) + \gamma(c_l)T]
\end{aligned} \tag{4.4}$$

Simplifying (4.4), we get,

$$E - E' = (\epsilon_1 + \epsilon_2)[(\mu(c_k) - \gamma(c_k)) - (\mu(c_l) - \gamma(c_l))]$$

If we say that in Case 2 the energy consumed is less, this means

$$\begin{aligned}
E - E' & > 0 \\
(\epsilon_1 + \epsilon_2)[(\mu(c_k) - \gamma(c_k)) - (\mu(c_l) - \gamma(c_l))] & > 0
\end{aligned}$$

$$(\mu(c_k) - \gamma(c_k)) > (\mu(c_l) - \gamma(c_l)) \quad (4.5)$$

(4.5) is in contradiction to our assumption. QED.  $\square$

We show which machines should be given comparatively more work than others. There is a special case in which the idle power of a machine is proportional to its working power. The following corollary covers machine precedence in such a case.

**Corollary 4.2.** *When the ratio between the working power consumption  $(\mu(c_i))$  and idle power consumption  $(\gamma(c_i))$  of the machine is some constant  $z, \forall i, 1 \leq i \leq m$ , then  $\mu(c_i) - \gamma(c_i)$  is proportional to  $\mu(c_i)$ , so we need to index the machines in the order of  $\mu(c_i)$ .*

For energy optimality it may well be suitable to give work to only some of the machines while letting others run completely idle. We now state the condition showing which machines should be used when using only a subset of all the machines is beneficial.

**Lemma 4.3.** *If we give work to only some  $r$  machines, where  $1 \leq r \leq m$ , then, for reduced energy consumption, these are the  $r$  machines that form the set  $\{c_1, c_2, \dots, c_r\}$  given our index order.*

*Proof.* For the  $r$  machines working,  $\tau(c_i) > 0$ , and for the other  $m - r$  machines  $\tau(c_i) = 0$ . If  $\tau(c_{r+1}) > 0$  then there must be any  $\tau(c_i)$  from the set  $\{\tau(c_1), \tau(c_2), \dots, \tau(c_r)\}$  which is equal to 0. But by Lemma 4.1,  $\tau(c_1) \geq \tau(c_2) \geq \tau(c_3) \geq \dots \geq \tau(c_r) > 0$  since  $\tau(c_{r+1}) > 0$ . These two statements are contradictory and so it is not possible that  $\tau(c_{r+1}) > 0$ . Hence it stands proved that for energy optimality, if we give work to only some  $r$  machines where  $1 \leq r \leq m$ , then these  $r$  machines are of the set  $\{c_1, c_2, \dots, c_r\}$ .  $\square$

Using Lemma 4.3, given some number of machines to be used, we can find which machines should be assigned jobs. We now state and prove the condition which decides how many machines should be used so that energy consumption of the system is minimal.

Let the energy consumption of a system of  $m$  machines, when jobs are distributed to  $r$  machines, be given by  $E_{m,r}$ . If we add one more machine to the working set of machines, then we take some amount of work from previously working machine(s) and give that amount of work to the new machine.  $s(c_i)$  represents the amount of work taken away from machine  $c_i$  and given to other machine(s).

**Theorem 4.4.** *When  $r - 1$  machines are working, for machine  $c_r$  to be given work (i.e.,  $\tau(c_r) \neq 0$ ) and result in reduced energy consumption, the following must hold.*

$$\sum_{i=1}^{r-1} [(\mu(c_i) - \gamma(c_i) - \mu(c_r) + \gamma(c_r))s(c_i)] + s(c_1) \sum_{i=1}^m \gamma(c_i) > 0 \quad (4.6)$$

*Proof.* We prove this by construction. If only  $r$  of the  $m$  machines are used and the rest remain idle all the time, (3.3) can be re-written as follows, where the energy is expressed by  $E_{m,r}$ :

$$E_{m,r} = \sum_{i=1}^{r-1} [\mu(c_i)\tau(c_i) + \gamma(c_i)(T - \tau(c_i))] \\ + \mu(c_r)\tau(c_r) + \gamma(c_r)(T - \tau(c_r)) + \sum_{i=r+1}^m \gamma(c_i)T$$

Putting  $T = \tau(c_1)$ , [by (3.5), Lemma 4.1 and Lemma 4.3]

$$E_{m,r} = \sum_{i=1}^{r-1} [\mu(c_i)\tau(c_i) + \gamma(c_i)(\tau(c_1) - \tau(c_i))] \\ + \mu(c_r)\tau(c_r) + \gamma(c_r)(\tau(c_1) - \tau(c_r)) + \sum_{i=r+1}^m \gamma(c_i)\tau(c_1) \quad (4.7)$$

Re-arranging (4.7),

$$E_{m,r} = \sum_{i=1}^{r-1} (\mu(c_i) - \gamma(c_i))\tau(c_i) + (\mu(c_r) - \gamma(c_r))\tau(c_r) + \tau(c_1) \sum_{i=1}^m \gamma(c_i) \quad (4.8)$$

Also, we know that,

$$W = \sum_{i=1}^r \tau(c_i) \quad (4.9)$$

Taking out the term  $r$  from (4.9),

$$W = \sum_{i=1}^{r-1} \tau(c_i) + \tau(c_r) \\ \tau(c_r) = W - \sum_{i=1}^{r-1} \tau(c_i) \quad (4.10)$$

Putting the value of  $\tau(c_r)$  from (4.10) in (4.8), we get,

$$E_{m,r} = \sum_{i=1}^{r-1} (\mu(c_i) - \gamma(c_i))\tau(c_i) + (\mu(c_r) \\ - \gamma(c_r))(W - \sum_{i=1}^{r-1} \tau(c_i)) + \tau(c_1) \sum_{i=1}^m \gamma(c_i) \quad (4.11)$$

Re-arranging (4.11),

$$E_{m,r} = \sum_{i=1}^{r-1} [(\mu(c_i) - \gamma(c_i) - \mu(c_r) + \gamma(c_r))\tau(c_i)] + (\mu(c_r) - \gamma(c_r))W + \tau(c_1) \sum_{i=1}^m \gamma(c_i) \quad (4.12)$$

This is a general equation for the energy of a system with  $m$  machines, where jobs are given to  $r$  machines, where  $1 \leq r \leq m$  and  $W$  is the total working time to complete all jobs.

If we give jobs to  $r - 1$  machines in a  $m$  machine system, its energy can be derived by putting  $r = r - 1$  in (4.12). Also the  $\tau(c_i)$ s are changed to  $\tau(c_i)'$ s as in the following:

$$E_{m,r-1} = \sum_{i=1}^{r-2} [\mu(c_i) - \gamma(c_i) - \mu(c_{r-1}) + \gamma(c_{r-1})]\tau(c_i)' + (\mu(c_{r-1}) - \gamma(c_{r-1}))W + \tau(c_1)' \sum_{i=1}^m \gamma(c_i) \quad (4.13)$$

It is better that we give work to more machines only when the energy consumption by doing so is lowered. Hence, if  $E_{m,r-1} > E_{m,r}$ , then only we should give work to  $r$  machines, else we give to  $r - 1$  machines only. Thus the condition for using  $r$  machines can be written as:

$$E_{m,r-1} - E_{m,r} > 0 \quad (4.14)$$

Using (4.12) and (4.13),

$$E_{m,r-1} - E_{m,r} = \sum_{i=1}^{r-1} [(\mu(c_i) - \gamma(c_i) - \mu(c_r) + \gamma(c_r))(\tau(c_i)' - \tau(c_i))] + (\tau(c_1)' - \tau(c_1)) \sum_{i=1}^m \gamma(c_i) \quad (4.15)$$

Taking  $\tau(c_i)' - \tau(c_i) = s(c_i)$ , we get,

$$E_{m,r-1} - E_{m,r} = \sum_{i=1}^{r-1} [(\mu(c_i) - \gamma(c_i) - \mu(c_r) + \gamma(c_r))s(c_i)] + s(c_1) \sum_{i=1}^m \gamma(c_i)$$

Here,  $s(c_i)$  informally signifies the amount of work taken away from machine  $c_i$ , to facilitate giving work to the machine  $c_r$ .

To give the jobs to  $r$  machines  $E_{m,r-1} - E_{m,r} > 0$ .

$$\sum_{i=1}^{r-1} [(\mu(c_i) - \gamma(c_i) - \mu(c_r) + \gamma(c_r))s(c_i)] + s(c_1) \sum_{i=1}^m \gamma(c_i) > 0$$

QED. □

Having derived the condition describing which machines to use for doing jobs, we now find out the amount of work to be given to each of these machines.

**Theorem 4.5.** *If we give jobs to  $r > 1$  machines, then for minimum energy consumption, the distribution of jobs on all  $r$  machines should be equal and given by  $\frac{W}{r}$ .*

*Proof.* We prove this by contradiction. Consider two cases, one in which the distribution of work among the machines, which qualify to work according to Theorem 4.4 is equal, and the other in which the work distribution is unequal. We claim, to show the contradiction, that in the case in which the distribution is equal, the energy consumption is greater as compared to the other, and that such is hence a non-optimal distribution.

Case 1: Equal distribution,  $\tau(c_i) = \frac{W}{r}, \forall i, 1 \leq i \leq r$ , i.e.,  $\tau(c_1) = \tau(c_2) = \tau(c_3) = \dots = \tau(c_r) = \frac{W}{r}$ .

Case 2: Unequal distribution,  $\tau(c_1) = \frac{W}{r} + \epsilon$ ,  $\tau(c_2) = \tau(c_3) = \dots = \tau(c_{r-1}) = \frac{W}{r}$  and  $\tau(c_r) = \frac{W}{r} - \epsilon$ .

Here we assign more work to machine 1 compared to that required by equal distribution of work among  $r$  machines (by Lemma 4.1, a bias has to favor smaller-numbered machines, and therefore machine 1 most of all). The amount of extra work given to machine 1 is  $\epsilon$  and this amount of work is withdrawn from machine  $c_r$ . We chose only to alter the work distributions of machines 1 and  $r$  to keep the proof simple. We could have chosen any machine  $c_k$  for doing extra work and any machine  $c_l$  for doing lesser work, where  $k > l$ . But for allotting extra work  $\epsilon$  to machine  $c_k$ , we have to increase the amount of work assigned to all machines in the set  $\{c_1, c_2, \dots, c_k\}$  by at least  $\epsilon$ , so that Lemma 4.1 is satisfied.

Similarly for reducing the work of machine  $c_l$  by  $\epsilon$ , we have to reduce the amount of work assigned to all machines in the set  $\{c_{l+1}, c_{l+2}, \dots, c_r\}$  by at least  $\epsilon$  so that Lemma 4.1 is satisfied. Hence if we want to alter the amount of work of just two machines (to keep the proof simple) from an equal distribution, then we have alter it for machine  $c_1$  and  $c_r$ . It may be noted that our proof is generalizable for imbalances involving any numbers of machines.

Let the energy in Case 1 be denoted by  $E'_{m,r}$  and the energy in Case 2 be  $E''_{m,r}$ . Then according to our proposition:

$$\begin{aligned} E''_{m,r} &< E'_{m,r} \\ E''_{m,r} - E'_{m,r} &< 0 \end{aligned} \tag{4.16}$$



Rewriting (4.12) after expanding, we get

$$\begin{aligned}
E_{m,r} = & (\mu(c_1) - \gamma(c_1) - \mu(c_r) + \gamma(c_r))\tau(c_1) \\
& + \sum_{i=2}^{r-1} [(\mu(c_i) - \gamma(c_i) - \mu(c_r) + \gamma(c_r))\tau(c_i)] \\
& + (\mu(c_r) - \gamma(c_r))W + \tau(c_1) \sum_{i=1}^m \gamma(c_i)
\end{aligned} \tag{4.17}$$

Putting the respective values of  $\tau(c_i)$ 's in (4.17) to derive  $E'_{m,r}$ ,

$$\begin{aligned}
E'_{m,r} = & (\mu(c_1) - \gamma(c_1) - \mu(c_r) + \gamma(c_r)) \left( \frac{W}{r} \right) \\
& + \sum_{i=2}^{r-1} \left[ (\mu(c_i) - \gamma(c_i) - \mu(c_r) + \gamma(c_r)) \left( \frac{W}{r} \right) \right] \\
& + (\mu(c_r) - \gamma(c_r))W + \left( \frac{W}{r} \right) \sum_{i=1}^m \gamma(c_i)
\end{aligned} \tag{4.18}$$

Similarly, we may derive  $E''_{m,r}$  as:

$$\begin{aligned}
E''_{m,r} = & (\mu(c_1) - \gamma(c_1) - \mu(c_r) + \gamma(c_r)) \left( \frac{W}{r} + \epsilon \right) \\
& + \sum_{i=2}^{r-1} \left[ (\mu(c_i) - \gamma(c_i) - \mu(c_r) + \gamma(c_r)) \left( \frac{W}{r} \right) \right] \\
& + (\mu(c_r) - \gamma(c_r))W + \left( \frac{W}{r} + \epsilon \right) \sum_{i=1}^m \gamma(c_i)
\end{aligned} \tag{4.19}$$

Subtracting (4.18) from (4.19), we get

$$E''_{m,r} - E'_{m,r} = \epsilon [(\mu(c_1) - \gamma(c_1) - \mu(c_r) + \gamma(c_r)) + \sum_{i=1}^m \gamma(c_i)] \tag{4.20}$$

Using (4.16) and (4.20), we get

$$(\mu(c_1) - \gamma(c_1) - \mu(c_r) + \gamma(c_r)) + \sum_{i=1}^m \gamma(c_i) < 0 \tag{4.21}$$

Now from Theorem 4.4,

$$\sum_{i=1}^{r-1} [(\mu(c_i) - \gamma(c_i) - \mu(c_r) + \gamma(c_r))s(c_i)] + s(c_1) \sum_{i=1}^m \gamma(c_i) > 0$$

Re-writing the above equation, we get,

$$\begin{aligned}
& (\mu(c_1) - \gamma(c_1) - \mu(c_r) + \gamma(c_r))s(c_1) + \sum_{i=1}^{r-1} [(\mu(c_i) \\
& - \gamma(c_i) - \mu(c_r) + \gamma(c_r))s(c_i)] + s(c_1) \sum_{i=1}^m \gamma(c_i) > 0
\end{aligned} \tag{4.22}$$

Now  $s(c_i) \geq 0, \forall i \in \{1, 2, \dots, r\}$  and according to our Assumption 4 in Section 3,  $(\mu(c_i) - \gamma(c_i) - \mu(c_r) + \gamma(c_r)) \leq 0, \forall i \in \{1, 2, \dots, r\}$ . Hence the first and the second term of (4.22) are negative, and the third term is positive. So for the condition in (4.22) to hold, the following is necessary:

$$(\mu(c_1) - \gamma(c_1) - \mu(c_r) + \gamma(c_r))s(c_1) + s(c_1) \sum_{i=1}^m \gamma(c_i) > 0$$

This means,

$$(\mu(c_1) - \gamma(c_1) - \mu(c_r) + \gamma(c_r)) + \sum_{i=1}^m \gamma(c_i) > 0. \tag{4.23}$$

But (4.21) is in contradiction with (4.23).

Hence,  $E''_{m,r} - E'_{m,r} < 0$  is false, which means,  $E'_{m,r} < E''_{m,r}$ . Hence it stands proved that we should give equal fractions of jobs to all the machines to get minimum energy. QED.  $\square$

Thus, for energy-minimal scheduling in this setting, work has to be divided equally among  $r$  machines, where  $r$  is chosen to satisfy (4.6). Using these results we may give a more general statement.

**Corollary 4.6.** *When a given set of machines are chosen for work, then all those machines should be in working state for an equal length of time for energy-minimal scheduling.*

In the next subsection, we analyze the case when the speeds of machines are also different. That case is more general and includes the current case of identical speeds, but the results for the identical speed case help develop theorems for variable speed system.

## 4.2 Systems with Different Speed Machines

In the previous subsection we gained insight on how a subset of machines can prove to be more advantageous for working for the entire duration of the makespan, while the rest of the machines are better left idle at all times. In this subsection we use the results of previous subsection to develop a theory for systems in which machines can have different speeds.

From Corollary 4.6, we know that even when the speeds of machines are different, all the machines of the working set should be active for the complete makespan. This means that if only a working set  $\mathcal{R}$  of machines is assigned work, then,

$$\tau(c_i) = \tau(c_j) = T, \forall i, j \in \mathcal{R} \quad (4.24)$$

$$\tau(c_k) = 0, \forall k \notin \mathcal{R} \quad (4.25)$$

Hence, if work is assigned to the subset  $\mathcal{R}$  of  $m$  machines, where  $|\mathcal{R}| = r$ , the energy consumption can be found using (4.24) and (4.25) in (3.3), we get,

$$E_{m,r} = \sum_{i \in \mathcal{R}} \mu(c_i)T + \sum_{i \notin \mathcal{R}} \gamma(c_i)T \quad (4.26)$$

$$= \left[ \sum_{i \in \mathcal{R}} (\mu(c_i) - \gamma(c_i)) + \sum_{i=1}^m \gamma(c_i) \right] T \quad (4.27)$$

Putting the value of  $\sum_{i=1}^m \gamma(c_i) = \Gamma$  in (4.27)

$$E_{m,r} = \left[ \sum_{i \in \mathcal{R}} (\mu(c_i) - \gamma(c_i)) + \Gamma \right] T \quad (4.28)$$

Using (4.24), the makespan of the system can be found as,

$$T = \frac{W}{\sum_{i \in \mathcal{R}} v(c_i)} \quad (4.29)$$

Since our aim is to find out relative distribution of work, without any loss of generality, we can assume that the total work is one unit to make the representation simpler, hence (4.29) can be re-written as:

$$T = \frac{1}{\sum_{i \in \mathcal{R}} v(c_i)} \quad (4.30)$$

Using (4.28) in (4.30) we get,

$$E_{m,r} = \frac{\sum_{i \in \mathcal{R}} (\mu(c_i) - \gamma(c_i)) + \Gamma}{\sum_{i \in \mathcal{R}} v(c_i)} \quad (4.31)$$

We now find out the set  $\mathcal{R}$  for which  $E_{m,r}$  of (4.31) is minimum. For the ease of representation, let

$$p_r = \sum_{i \in \mathcal{R}} (\mu(c_i) - \gamma(c_i)) + \Gamma \quad (4.32)$$

and

$$q_r = \sum_{i \in \mathcal{R}} v(c_i) \quad (4.33)$$

Hence (4.31) can be written as

$$E_{m,r} = \frac{p_r}{q_r}$$

We now find out that if all the work has to be assigned to only one machine, then which will that machine be. We present the answer in our first lemma of this section.

**Lemma 4.7.** *If all the work is assigned to one machine  $c_i$ , i.e.,  $|\mathcal{R}| = 1$  and  $\mathcal{R} = \{c_i\}$ , then for energy minimality this should be the machine with minimum  $\frac{\mu(c_i) - \gamma(c_i) + \Gamma}{v(c_i)}$ .*

*Proof.* Substituting  $\mathcal{R} = \{c_i\}$  in (4.31), we get,

$$E_{m,1} = \frac{(\mu(c_i) - \gamma(c_i)) + \Gamma}{v(c_i)}$$

Hence to minimize energy consumption  $E_{m,1}$ , we need to choose  $c_i$  for which  $\frac{\mu(c_i) - \gamma(c_i) + \Gamma}{v(c_i)}$  is minimum.  $\square$

As mentioned in Assumption 4 in Section 3, we order the machines such that the first machine is with minimum  $\frac{\mu(c_i) - \gamma(c_i) + \Gamma}{v(c_i)}$ . Hence here  $i = 1$ , and the first machine is given work in this case.

Now, if we want to give job to two machines then we will only give if the energy consumed by two machines is less than the energy consumption of one machine. The same principle applies whenever we want to expand the working set of machines. The new expanded set must have lesser energy consumption than the previous set. In the next theorem, we specify the conditions under which a set can be expanded.

**Theorem 4.8.** *If a machine  $c_j$  has to be included in the working set of machines  $\mathcal{R}$ , then the following condition must be satisfied*

$$\frac{\sum_{i \in \mathcal{R}} (\mu(c_i) - \gamma(c_i)) + \Gamma}{\sum_{i \in \mathcal{R}} v(c_i)} > \frac{\mu(c_j) - \gamma(c_j)}{v(c_j)} \quad (4.34)$$

*Proof.* Re-writing our condition (4.34), using (4.32) and (4.33)

$$\frac{p_r}{q_r} > \frac{\mu(c_j) - \gamma(c_j)}{v(c_j)}$$

Multiplying the denominators both sides,

$$p_r v(c_j) > \mu(c_j) q_r - \gamma(c_j) q_r$$

Adding  $p_r q_r$  both sides,

$$\begin{aligned} p_r v(c_j) + p_r q_r &> \mu(c_j) q_r - \gamma(c_j) q_r + p_r q_r \\ p_r (v(c_j) + q_r) &> q_r (\mu(c_j) - \gamma(c_j) + p_r) \\ \frac{p_r}{q_r} &> \frac{p_r + \mu(c_j) - \gamma(c_j)}{q_r + v(c_j)} \end{aligned}$$

Back substituting  $p_r$  and  $q_r$ , we get

$$\frac{\sum_{i \in \mathcal{R}} (\mu(c_i) - \gamma(c_i)) + \Gamma}{\sum_{i \in \mathcal{R}} v(c_i)} > \frac{\sum_{i \in \mathcal{R}} (\mu(c_i) - \gamma(c_i)) + \Gamma + \mu(c_j) - \gamma(c_j)}{\sum_{i \in \mathcal{R}} v(c_i) + v(c_j)}$$

Let  $\mathcal{R}'$  be the new set formed for including  $c_j$  in  $\mathcal{R}$ .

$$\frac{\sum_{i \in \mathcal{R}} (\mu(c_i) - \gamma(c_i)) + \Gamma}{\sum_{i \in \mathcal{R}} v(c_i)} > \frac{\sum_{i \in \mathcal{R}'} (\mu(c_i) - \gamma(c_i)) + \Gamma}{\sum_{i \in \mathcal{R}'} v(c_i)} \quad (4.35)$$

The left hand side of (4.35) gives the energy consumption by working set  $\mathcal{R}$ . The right hand side of (4.35), gives the energy consumption when machine  $c_j$  is included in the original working set. Hence the energy consumption of the new set is lesser when the condition (4.34) is satisfied.  $\square$

The previous theorem gives the condition in which a machine could be included in the working set to reduce the energy consumption of the system. But there can be many machines which satisfy this condition. We need to find out the machine which reduces the energy consumption by largest amount and hence make the system energy minimal. Our next theorem specifies which machine should be given preference for inclusion in working set.

**Theorem 4.9.** *Given any two machines  $c_j$  and  $c_k$  which qualify to be included in working set  $\mathcal{R}$  according to Theorem 4.8, i.e.,*

$$\frac{\mu(c_j) - \gamma(c_j)}{v(c_j)} < \frac{p_r}{q_r} \quad (4.36)$$

and

$$\frac{\mu(c_k) - \gamma(c_k)}{v(c_k)} < \frac{p_r}{q_r} \quad (4.37)$$

*For minimal energy consumption, machine  $c_j$  shall be chosen over  $c_k$  if*

$$\frac{\mu(c_j) - \gamma(c_j)}{v(c_j)} < \frac{\mu(c_k) - \gamma(c_k)}{v(c_k)} \quad (4.38)$$

*Proof.* Given the condition (4.38), we would like to derive energy equations from it. Multiplying the denominators both sides in (4.38), we get,

$$(\mu(c_j) - \gamma(c_j))v(c_k) < (\mu(c_k) - \gamma(c_k))v(c_j)$$

By adding and subtracting  $(\mu(c_j) - \gamma(c_j))v(c_j)$  in equation (4.2), we get,

$$\frac{(\mu(c_j) - \gamma(c_j)) - (\mu(c_k) - \gamma(c_k))}{v(c_j) - v(c_k)} < \frac{(\mu(c_j) - \gamma(c_j))}{v(c_j)} \quad (4.39)$$

From (4.36) and (4.39). We get,

$$\frac{(\mu(c_j) - \gamma(c_j)) - (\mu(c_k) - \gamma(c_k))}{v(c_j) - v(c_k)} < \frac{(\mu(c_j) - \gamma(c_j))}{v(c_j)} < \frac{p_r}{q_r}$$

$$\frac{(\mu(c_j) - \gamma(c_j)) - (\mu(c_k) - \gamma(c_k))}{v(c_j) - v(c_k)} < \frac{p_r}{q_r}$$

$$((\mu(c_j) - \gamma(c_j)) - (\mu(c_k) - \gamma(c_k)))q_r < p_r(v(c_j) - v(c_k))$$

$$(\mu(c_j) - \gamma(c_j))q_r - (\mu(c_k) - \gamma(c_k))q_r - p_r v(c_j) + p_r v(c_k) < 0 \quad (4.40)$$

Adding (4.2) and (4.40), we get,

$$\begin{aligned} & (\mu(c_j) - \gamma(c_j))q_r - (\mu(c_k) - \gamma(c_k))q_r - p_r v(c_j) + p_r v(c_k) \\ & + (\mu(c_j) - \gamma(c_j))v(c_k) - (\mu(c_k) - \gamma(c_k))v(c_j) < 0 \end{aligned} \quad (4.41)$$

Now adding and subtracting  $p_r q_r$  in (4.41), we get,

$$\frac{p_r + \mu(c_j) - \gamma(c_j)}{q_r + v(c_j)} < \frac{p_r + \mu(c_k) - \gamma(c_k)}{q_r + v(c_k)} \quad (4.42)$$

In (4.42), left hand side gives the energy consumption of the system when machine  $c_j$  is included in the working set  $\mathcal{R}$ , while the right hand side gives the energy consumption of the system when machine  $c_k$  is included. The energy consumption is lesser with machine  $c_j$  in comparison to machine  $c_k$  and hence machine  $c_j$  would be given preference to be included in working set.  $\square$

When we put Lemma 4.7 and Theorem 4.9 together, we get the order in which machines will get preference to be allotted work. This order is same as mentioned in Assumption 4 in Section 3. Combining Lemma 4.7, Theorem 4.9 and Assumption 4, we can state a Corollary.

**Corollary 4.10.** *If the machines are indexed in an order such that the first machine is the one with smallest  $\frac{\mu(c_i) - \gamma(c_i) + \Gamma}{v(c_i)}$  and afterwards in increasing order of  $\frac{\mu(c_i) - \gamma(c_i)}{v(c_i)}$ , and if the working set contains  $r$  machines, then for minimal energy consumption  $\mathcal{R} = \{c_1, c_2, \dots, c_r\}$ , where each machine in this set is working for an equal amount of time.*

Using this Corollary, (4.31) can be re-written as,

$$E_{m,r} = W \left[ \frac{\sum_{i=1}^r (\mu(c_i) - \gamma(c_i)) + \Gamma}{\sum_{i=1}^r v(c_i)} \right] \quad (4.43)$$

which gives the energy consumption of a system with total work  $W$ .

From (4.43), energy is dependent on work  $W$ , but from (4.23) and Theorems 4.8 and 4.9 we know that finding the number of working machines (value of  $r$ ) from the given set  $\mathcal{C}$  of machines is independent of work  $W$ .

We see that (4.23), (4.34) and (4.38) are independent of  $W$ , showing that the machines used and the scheduling decisions do not depend upon the quantum of the work given to the system. This in turn means that, some systems can be inherently wasteful and energy inefficient.

These results can be used in configuring the system, that which machines we want in system model. In some cases we do not want certain machines in the system. When initially we have certain machines (set  $\mathcal{C}$ ), and we have to design the system, then using our results we can choose which machines we want in our system.

**Corollary 4.11.** *Given a set  $\mathcal{C}$  of machines, we use  $\{c_1, c_2, \dots, c_r\}$  in our system and leave  $\{c_{r+1}, c_{r+2}, \dots, c_m\}$  machines out of system.*

Our results are independent of the work to be done. From results we know that, to minimize the energy of the system some machines should always idle. So these results (which guide which machines should be used), will help in designing of the system. So that we can shut-off some machines which are always kept idle.

In this section, we found the precedence amongst machines for allotting work based on their energy and speed specifications. The results derived in this section are applicable for handling any class of jobs, which means that the precedence of machines will remain same irrespective of whether the jobs are divisible, non-divisible or whether they have precedence constraints. In the next sections we consider different classes of systems based on type of jobs.

## 5 Algorithms and Complexity

Depending upon the jobs to be executed the systems can be classified into many categories. We have already mentioned those categories in Section 3. In the current section we analyze the complexity of scheduling problem for these various types of systems and give scheduling algorithms for the same. We first analyze systems with identical speed machines and then move ahead to systems with different speeds.

## 5.1 Systems with Identical Speed Machines

In Section 4.1, we gave results which govern the scheduling of systems with identical speed machines. Given the total amount of work and energy specifications of machines, we can find the number of machines which have to be assigned work using results of Section 4.1. Algorithm 1 is an implementation of those results.

---

**Algorithm 1:** Find the number of working machines of the system in order to minimize the energy consumption when the speeds of machines are identical

---

```

input : Number of machines ( $m$ ), working power of machines ( $\mu(c_i)$ ), idle power of machines
         ( $\gamma(c_i)$ ), total work to be done ( $W$ )
output: Number of working machines ( $r$ ), makespan ( $T$ ), energy ( $E$ )

1 for  $i = 1$  to  $m$  do
2   | calculate  $\mu(c_i) - \gamma(c_i)$ 
3 end
4 sort ( $\mu(c_i) - \gamma(c_i)$ );
5  $low \leftarrow 1$ ,  $high \leftarrow m$ ;
6 while  $low \leq high$  do
7   |  $mid \leftarrow low + \frac{(high - low)}{2}$ ;
8   |  $E(k) \leftarrow (\frac{W}{k})[\sum_{i=1}^k \mu(c_i) + \sum_{i=k+1}^m \gamma(c_i)]$ ;
9   | Calculate  $E(mid - 1)$ ,  $E(mid)$ ,  $E(mid + 1)$ ;
10  | if ( $E(mid - 1) > E(mid)$ ) && ( $E(mid) \leq E(mid + 1)$ ) then
11  |   |  $r \leftarrow mid$ ;
12  |   | return  $r$ ;
13  | else
14  |   | if  $E(mid - 1) > E(mid)$  then
15  |   |   |  $low \leftarrow mid$ ;
16  |   | else
17  |   |   |  $high \leftarrow mid$ ;
18  |   | end
19  | end
20 end
21  $T \leftarrow \frac{W}{r}$ ;
22 return  $T$ ;

```

---

Algorithm 1 takes the number of machines, the working power and idle power of the machines, and the total work to be done as input, and gives the value of  $r$ , i.e., the number of working machines, as output. The machines are indexed in the precedence order given by Assumption 4 in Section 3. Now, we apply binary search to find the value of  $r$ , such that we get minimal energy value of the system. The algorithm is based on the previously given results. The computation complexity of Algorithm 1 is  $\mathcal{O}(m)$ . We now find the complexity of scheduling problems and give algorithms for divisible and non-divisible jobs.



### 5.1.1 Divisible Loads

In this case we are given a set  $\mathcal{P}$  of jobs, where jobs can be arbitrarily broken into any number of smaller jobs and these smaller jobs can be executed parallelly on different machines. We consider the energy consumption overhead due to division/breaking of jobs overhead to be null. Since the jobs are divisible, they can be trivially distributed over  $r$  machines with makespan  $T$ , where  $r$  and  $T$  are given by Algorithm 1.

Hence the energy of system with divisible loads can be calculated using following equation.

$$E_{m,r} = T \left[ \sum_{i=1}^r \mu(c_i) + \sum_{i=r+1}^m \gamma(c_i) \right] \quad (5.1)$$

Using Algorithm 1 and (5.1) we can find the energy of the system.

**Remark 5.1.** *Energy-minimal scheduling of divisible jobs on identical speed machines can be done in linear time.*

We now analyze the complexity of scheduling of non-divisible Loads.

### 5.1.2 Non-divisible Loads

Given the set  $\mathcal{P}$  of non-divisible jobs, and time required to execute job  $p_j$  on a machine with unit speed  $\psi(p_j)$ ; our aim is to distribute the set  $\mathcal{P}$  of jobs among the given set  $\mathcal{C}$  of machines such that energy consumption is minimum. From Corollary 4.6 it is evident that whichever machines are chosen to work, they should be working for an equal amount of time to achieve energy minimality. But it is not straightforward of sometimes even possible to distribute work in such a way when jobs are non-divisible. We prove that it is an NP-hard problem to calculate the minimal-energy schedule for the current case.

**Proposition 5.2.** *In a system where machines have identical speeds and jobs are non-divisible, computing the energy-minimal schedule is an NP-hard problem.*

*Proof.* Let  $r$  denote the number of machines which are given work and  $T$  is the makespan according to Algorithm 1. Since all the machines have identical speed, the scheduling problem is to make exclusive subsets  $P_i$  from set  $\mathcal{P}$  such that the sum of work in set  $i$  is  $T$ , i.e.,

$$\sum_{j \in P_i} \psi(p_j) = T \quad \forall i; 1 \leq i \leq r \quad (5.2)$$

Given a finite set of positive numbers and another positive number called the goal, to find the subset whose sum is closest to the goal is well known as the classical *subset-sum problem* (Johnson, 1973). In the current scenario, the set of positive numbers  $\{\psi(p_j) : j \in \mathcal{P}\}$  has to be distributed in subsets  $P_i$  such that sum of each subset is the positive number

$T$ . For each subset  $(P_i ; i \leq r)$ , this problem of finding the exclusive subsets  $P_i$  can be seen as the classical *subset-sum problem*. Since the subset-sum problem is NP-hard (Johnson, 1973), and our problem is reduced to subset-sum problem; energy-minimal scheduling of jobs in a system where jobs are non-divisible is NP-hard.  $\square$

Since finding the energy-minimal schedule is NP-hard, we develop an approximation algorithm to find energy efficient schedules. Though we strive to achieve the makespan given by Algorithm 1, in a system with non-divisible loads it is not always possible to achieve it. We state the theorem which specifies the ideal makespan  $T_o$  and working set of machines  $\mathcal{R}_o$  for energy minimality of system with non-divisible jobs.

**Lemma 5.3.** *Given a system with identical speed machines and a set  $\mathcal{P}$  of jobs with longest job of length  $\psi(p_{max})$ , the makespan is given by  $T_o = \max(T, \psi(p_{max}))$  and the working set of machines  $\mathcal{R}_o = \{c_1, c_2, \dots, c_{r_o}\}$ , where*

$$r_o = \left\lceil \frac{W}{T_o} \right\rceil \quad (5.3)$$

*Proof.* It is not possible to have makespan lesser than  $\psi(p_{max})$  since jobs cannot be divided. Also, if we take  $T_o$  lesser than  $T$ , then we cannot have a energy-minimal schedule. Hence for energy minimality,

$$T_o = \max(T, \psi(p_{max})) \quad (5.4)$$

From Corollary 4.6, it is evident that work should be distributed equally if possible to the working machines. Hence with a possibly increased makespan, the number of working machines might get decreased and given by (5.3). Also from Lemma 4.3, it is evident that the working set of machines is given by:

$$\mathcal{R}_o = \{c_1, c_2, \dots, c_{r_o}\} \quad (5.5)$$

$\square$

It is not always possible to keep each machine working till time  $T_o$ . Some machines might work for more than  $T_o$  and some might get less than that because of non-divisibility of jobs. From Lemma 4.1, we know that the amount of work should be in decreasing order by the machine's indices. With all this information, the approximation algorithm for energy-minimal scheduling of non-divisible jobs can be given as follows.

In Algorithm 2, we index the machines in non-decreasing order of the difference of their working power and idle power. Jobs are indexed in non-increasing order of their weight. Initially we create  $r$  empty buckets for jobs to be filled in. First  $r$  jobs are chosen and given to each  $r$  bucket, such that each bucket has one element. Now, to distribute remaining  $n - r$  jobs to the buckets we follow a sequential process. The weighted sum of jobs of each bucket is calculated and the next job is included in the bucket in which the value is least. The same process is repeated until all jobs are included in one or other bucket or sets. Now

---

**Algorithm 2:** Approximation algorithm for energy-efficient scheduling of the system when the speeds of machines are identical

---

**input** : Number of machines ( $m$ ), working power of machines ( $\mu(c_i)$ ), idle power of machines ( $\gamma(c_i)$ ), total work to be done ( $W$ ),  $r_o$ ,  $T_o$   
**output**: Makespan ( $T$ ), energy ( $E$ )

```

1   $S = \{\}$  ;
2  for  $i = 1$  to  $m$  do
3     $d_i \leftarrow \mu(c_i) - \gamma(c_i)$  ;
4     $S = S \cup \{d_i\}$  ;
5  end
6  sort  $S$ ;
7   $J = \{\}$  ;
8  for  $j = 1$  to  $n$  do
9     $J = J \cup \{\psi(p_j)\}$  ;
10 end
11 reverse-sort ( $J$ );
12  $r \leftarrow r_o$ ;
13  $P_j = \{\}$ ;
14 for  $j = 1$  to  $r$  do
15    $P_j = P_j \cup \{p_j\}$ ;
16    $\sum_{j \in P_j} \psi(p_j) = b_j$ ;
17 end
18 for  $l = r + 1$  to  $n$  do
19   calculate  $\min(b_j)$ ;
20    $b_u \leftarrow \min(b_j)$ ;
21    $u \leftarrow j$  ;
22    $P_j = P_j \cup \{p_l\}$ ;
23    $\sum_{j \in P_j} \psi(p_j) = b_j$ ;
24 end
25 for  $i = 1$  to  $r$  do
26   assign  $p_i$  to  $c_i$  ( $\forall p_i \in P_i$ ) ;
27 end
28 for  $i = 1$  to  $r$  do
29    $\sum_{j \in P_j} \psi(p_j) = \tau(c_i)$ ;
30   sort  $\tau(c_i)$ ;
31    $T \leftarrow \max(\tau(c_i))$ ;
32    $\kappa(c_i) \leftarrow T - \tau(c_i)$ ;
33 end
34  $T \leftarrow \max \tau(c_i)$  ( $\forall p_i \in P_i$ ) ;
35 return  $T$ ;
36  $E \leftarrow \sum_{i=1}^r \mu(c_i) \tau(c_i) + T \sum_{i=r+1}^m \gamma(c_i)$ ;
37 return  $E$ ;

```

---

we index the buckets in non-decreasing order of their total weights. We assign the jobs of set  $i$  to machine  $i$ . Now the working time of all the machines is calculated. Hence we can calculate the makespan, which is the maximum working time of all the machines.

This approximation algorithm is inspired by Graham's algorithm (Graham, 1969) which arranges a set of independent non-divisible jobs on identical speed machines such that the makespan is minimum. We made some changes in the algorithm to mould it for different power specification of machines. We now calculate the bound on deviation from ideal energy consumption due to approximation.

**Theorem 5.4.** *The maximum possible ratio of energy consumption using Algorithm 2 and ideal energy consumption is given by:*

$$\frac{E^*_{max}}{E_o} = 1 + \frac{(\frac{4}{3} - \frac{1}{3r_o} - 1)\Gamma}{\sum_{i=1}^{r_o}(\mu(c_i) - \gamma(c_i)) + \Gamma} \quad (5.6)$$

*Proof.* From (3.3), if all the  $r_o$  machines are working for an equal amount of time  $T_o$ , then energy consumption is given by:

$$E_o = T_o \left[ \sum_{i=1}^{r_o} (\mu(c_i) - \gamma(c_i)) + \Gamma \right] \quad (5.7)$$

From Algorithm 2, the energy consumed is given by:

$$E^* = \sum_{i=1}^{r_o} (\mu(c_i) - \gamma(c_i)) \tau^*(c_i) + \Gamma T^* \quad (5.8)$$

Assume that the algorithm gave such  $\tau^*_i$  that all the machines other than  $c_j$  and  $c_k$  are allotted work equal to  $T_o$ . Mathematically,

$$\tau^*(c_j) = T_o + \epsilon, \quad \tau^*(c_k) = T_o - \epsilon \quad (5.9)$$

where  $j < k \leq r_o$ . And

$$\tau^*(c_i) = T_o, \quad \forall i \neq j, k, \quad i \leq r_o \quad (5.10)$$

Since  $j < k$ , according to our algorithm  $\tau^*(c_j) > \tau^*(c_k)$  and so  $\epsilon \geq 0$ . Using these values from (5.9) and (5.10) in (5.8), we get,

$$\begin{aligned} E^* &= (\mu(c_j) - \gamma(c_j))(T_o + \epsilon) + (\mu(c_k) - \gamma(c_k))(T_o - \epsilon) \\ &\quad + \sum_{i \neq j, k} (\mu(c_i) - \gamma(c_i))T_o + \Gamma T^* \end{aligned} \quad (5.11)$$

Simplifying above,

$$E^* = (\mu(c_j) - \gamma(c_j) - (\mu(c_k) - \gamma(c_k)))\epsilon + \sum_{i=1}^{r_o} (\mu(c_i) - \gamma(c_i))T_o + \Gamma T^* \quad (5.12)$$

Now since  $j < k$ , we have  $\mu(c_j) - \gamma(c_j) - (\mu(c_k) - \gamma(c_k)) \leq 0$ . To compute the bound we are looking for the case where  $E^*$  is maximum. That will occur when

$$\mu(c_j) - \gamma(c_j) - (\mu(c_k) - \gamma(c_k)) = 0 \quad (5.13)$$

Using (5.13) in (5.12), we get,

$$E^*_{max} = \sum_{i=1}^{r_o} (\mu(c_i) - \gamma(c_i)) T_o + \Gamma T^* \quad (5.14)$$

Now according to (Graham, 1969), the bound on makespan is given by:

$$\frac{T^*}{T_o} = \frac{4}{3} - \frac{1}{3r_o} \quad (5.15)$$

Using this bound in (5.14), we get

$$E^*_{max} = \sum_{i=1}^{r_o} (\mu(c_i) - \gamma(c_i)) T_o + \Gamma T_o \left( \frac{4}{3} - \frac{1}{3r_o} \right) \quad (5.16)$$

Simplifying above

$$E^*_{max} = T_o \left( \sum_{i=1}^{r_o} (\mu(c_i) - \gamma(c_i)) + \Gamma \left( \frac{4}{3} - \frac{1}{3r_o} \right) \right) \quad (5.17)$$

$$E^*_{max} = T_o \left( \sum_{i=1}^{r_o} (\mu(c_i) - \gamma(c_i)) + \Gamma \right) + T_o \Gamma \left( \frac{4}{3} - \frac{1}{3r_o} - 1 \right) \quad (5.18)$$

Substituting the values, we get,

$$\frac{E^*_{max}}{E_o} = \frac{T_o (\sum_{i=1}^{r_o} (\mu(c_i) - \gamma(c_i)) + \Gamma) + T_o \Gamma (\frac{4}{3} - \frac{1}{3r_o} - 1)}{T_o [\sum_{i=1}^{r_o} (\mu(c_i) - \gamma(c_i)) + \Gamma]} \quad (5.19)$$

$$\frac{E^*_{max}}{E_o} = 1 + \frac{\Gamma (\frac{4}{3} - \frac{1}{3r_o} - 1)}{\sum_{i=1}^{r_o} (\mu(c_i) - \gamma(c_i)) + \Gamma} \quad (5.20)$$

□

In the worst case, the values of working power and idle power are equal, i.e.  $\mu(c_i) = \gamma(c_i); \forall i$ , so the bound is given by,

$$\begin{aligned} \frac{E^*_{max}}{E_o} &= 1 + \frac{\Gamma (\frac{4}{3} - \frac{1}{3r_o} - 1)}{\sum_{i=1}^{r_o} (\mu(c_i) - \gamma(c_i)) + \Gamma} \\ &= 1 + \frac{\Gamma (\frac{4}{3} - \frac{1}{3r_o} - 1)}{\Gamma} \\ &= 1 + \left( \frac{4}{3} - \frac{1}{3r_o} - 1 \right) \\ &= \frac{4}{3} - \frac{1}{3r_o} \end{aligned}$$

If  $r_o$  is very large, i.e. when  $r_o \rightarrow \infty$ , then,

$$\frac{E^*_{max}}{E_o} = \frac{4}{3} \quad (5.21)$$

(5.20) gives the lower bound of inefficiency of our algorithm and (5.21) gives the upper bound.

We now move ahead to analyze the class of scheduling problems in which machines can have different working speeds.

## 5.2 Systems with Different Speed Machines

In this section, along with power specifications of machines we also consider the speeds of machines to be different and give algorithms for both divisible jobs and non-divisible jobs. In Section 4.2, we proved various lemmas and theorems which govern the scheduling of systems with different speed machines. Using these results Algorithm 3 finds the set of machines which should be assigned work for the energy-minimal scheduling when the machines of the system have different speeds.

Algorithm 3 takes the number of machines, the working power, idle power and speeds of the machines, and the total work to be done as input, and gives the value of  $r$ , i.e., the number of working machines, as output. In the algorithm, we first index the machines in their precedence order, as given in Assumption 4 in Section 3. Then to calculate the value of  $r$ , we check the condition in (4.34) from Theorem 4.8. Then from the given values of total work to be done and the computed value of number of machines, we calculate the makespan. The algorithm also computes the amount of work that has to be given to respective machines. The complexity of Algorithm 3 is  $\mathcal{O}(m)$ .

We now find the complexity of scheduling problems and give algorithms for divisible and non-divisible jobs.

### 5.2.1 Divisible Loads

In this case, jobs can be arbitrarily broken into any number of smaller jobs and these smaller jobs can be executed parallelly on different machines. Also we consider the energy consumption overhead due to division/breaking of jobs overhead to be nil. In this scenario the jobs can be trivially distributed over  $r$  machines such that machine  $c_i$  gets  $w(c_i)$  amount of work (as specified by Algorithm 3). The energy consumption of system with divisible load can be given by:

$$E = \sum_{i=1}^m \left[ \mu(c_i) \frac{w(c_i)}{v(c_i)} + \gamma(c_i) \left( T - \frac{w(c_i)}{v(c_i)} \right) \right] \quad (5.22)$$

Clearly the scheduling of divisible jobs can be done by using Algorithm 3 which is a linear time algorithm.

---

**Algorithm 3:** Find the working machines and makespan of the system when the speeds of machines are different

---

**input** : Number of machines ( $m$ ), working power of machines ( $\mu(c_i)$ ), idle power of machines ( $\gamma(c_i)$ ), sum of idle power of all the machines ( $\Gamma$ ), speed of machines ( $v(c_i)$ ), total work to be done ( $W$ )

**output:** Number of working machines ( $r$ ), makespan ( $T$ )

```

1 for  $i = 1$  to  $m$  do
2   | calculate  $\frac{\mu(c_i) - \gamma(c_i) + \Gamma}{v(c_i)}$ 
3 end
4  $\frac{\mu(c_1) - \gamma(c_1) + \Gamma}{v(c_1)} \leftarrow \min(\frac{\mu(c_i) - \gamma(c_i) + \Gamma}{v(c_i)})$ ;
5 for  $i = 2$  to  $m$  do
6   | calculate  $\frac{\mu(c_i) - \gamma(c_i)}{v(c_i)}$ 
7 end
8 sort  $(\mu(c_i) - \gamma(c_i))$ ;
9 for  $r = 1$  to  $m$  do
10  |  $E(r) \leftarrow \lceil \frac{\sum_{i=1}^r (\mu(c_i) - \gamma(c_i) + \Gamma)}{\sum_{i=1}^r v(c_i)} \rceil W$ ;
11  | if  $\frac{a_{r+1} - b_{r+1}}{c_{r+1}} < E(r)$  then
12    |  $r \leftarrow r + 1$ ;
13    | return  $r$ ;
14  | else
15    | stop;
16  | end
17 end
18  $T \leftarrow \frac{W}{r}$ ;
19 return  $T$ ;
20  $w(c_i) \leftarrow T * v(c_i)$ ;
21 return  $w(c_i)$ ;

```

---

**Remark 5.5.** *Energy-minimal scheduling of divisible jobs on a system of machines running at different speeds can be done in linear time.*

Many might argue that many systems do not have the luxury to divide loads arbitrarily, but it must be noted that efficiency in a system with divisible loads can be seen as upper bound to the achievable efficiency. We proceed to analyze the case in which jobs are non-divisible in the next section.

### 5.2.2 Non-divisible Loads

In this section we first prove that scheduling non-divisible jobs for energy optimality on a system with machines having different speeds is an NP-hard problem. We then give an approximation algorithm for the same.

**Theorem 5.6.** *In a system where machines have different speeds and jobs are non-divisible; computing energy-minimal schedule in that case is an NP-hard problem.*

*Proof.* The amount of work  $w(c_i)$  that has to be assigned to machine  $c_i$  is given by Algorithm 3. Now we need to make exclusive subsets  $P_i$  from set  $\mathcal{P}$  such that the sum of work in set  $P_i$  is  $w(c_i)$ , i.e.,

$$\sum_{j \in P_i} \psi(p_j) = w(c_i) \quad \forall i; 1 \leq i \leq r \quad (5.23)$$

As also discussed with Theorem 5.2, this is in the form of the subset-sum problem which is NP-hard (Johnson, 1973).  $\square$

Since finding the energy-minimal schedule is NP-hard, we develop an approximation algorithm to find energy efficient schedules. The approximation algorithm tries to distribute jobs such that the ideal makespan specified by Algorithm 3 is achieved.

**Lemma 5.7.** *Given a system of machines with different speed and a set  $\mathcal{P}$  of jobs with longest job of length  $\psi(p_{max})$  and the speed of the fastest machine is given by  $v_{max}$ , the best possible achievable makespan is given by:*

$$T_o = \max(T, \frac{\psi_{max}}{v_{max}}) \quad (5.24)$$

where  $T$  is given by Algorithm 3.

*Proof.* If there is a job which cannot be completed within  $T$ , irrespective of the arrangement of jobs, then we increase the best achievable makespan  $T_o$  to accommodate that biggest job. The biggest job cannot be completed any earlier than  $\frac{\psi_{max}}{v_{max}}$ . Also, since any schedule with makespan lesser than  $T$  should be suboptimal. Hence,

$$T_o = \max(T, \frac{\psi_{max}}{v_{max}})$$

$\square$



If the makespan is increased then there can be possible reduction in number of working machines. The ideal number of working machines  $r_o$  is given by the minimum number of machines which satisfy,

$$\frac{W}{\sum_{i=1}^{r_o} v(c_i)} \geq T_o \quad (5.25)$$

Using this  $r_o$  as input, we present an approximation algorithm which tries to distribute jobs amongst machines with different speed such that the machines would work for an equal amount of time.

---

**Algorithm 4:** Approximation algorithm for energy-efficient scheduling of the system when the speeds of machines are different

---

**input** : Number of machines ( $m$ ), working power of machines ( $\mu(c_i)$ ), idle power of machines ( $\gamma(c_i)$ ), speed of machines ( $v(c_i)$ ), total work to be done ( $W$ ),  $r_o$ ,  $T_o$   
**output**: Makespan ( $T$ ), energy ( $E$ )  
**output**: Schedule

```

1 for  $i = 1$  to  $m$  do
2    $A(i) \leftarrow \frac{\mu(c_i) - \gamma(c_i)}{v(c_i)}$  ;
3 end
4 sort  $A(i)$  in non-decreasing order;
5 for  $j = 1$  to  $n$  do
6    $B(j) \leftarrow \psi(p_j)$  ;
7 end
8 sort  $B(j)$  in non-increasing order ;
9  $temp_i \leftarrow 0$  ;
10 for  $j = 1$  to  $n$  do
11   for  $i = 1$  to  $r_o$  do
12      $temp_i \leftarrow \frac{\psi(p_j)}{v_i} + temp_i$  ;
13   end
14   assign  $p_j$  to  $c_i$ , where  $c_i$  has  $\min(temp_i)$  ;
15 end
16  $T \leftarrow \max \tau(c_i)$ ;
17 return  $T$ ;
18  $E \leftarrow \sum_{i=1}^m [\mu(c_i) \frac{w(c_i)}{v(c_i)} + \gamma(c_i)(T - \frac{w(c_i)}{v(c_i)})]$ ;
19 return  $E$ ;

```

---

In Algorithm 4, we first index the machines in decreasing order of the speed of the machines. We also index the jobs in non-increasing order of their weights. Now, we take a job at a time and find that on which machine it will finish first, based on the speed of the machine. Then we assign that job to that particular machine. Similarly the process is repeated till all the jobs are assigned to some machines. Now, the working time of all machines is calculated, and the maximum working time of all machines is the makespan.

The approximation algorithm may not achieve the exact  $T_o$ , but it is within certain bounds.

The bound for the makespan given by Dobson (1984) of such an LPT (largest processing time) schedule is

$$\frac{19}{12} \quad (5.26)$$

Given this bound on makespan, we derive the bound on energy consumption of our algorithm.

**Theorem 5.8.** *When the speeds as well as the power specifications of the machines are different, and when jobs are non-divisible, then a bound on the energy consumption using Algorithm 4 and ideal energy consumption is given by:*

$$\frac{E^*_{max}}{E_o} \leq \frac{19}{12} \quad (5.27)$$

*Proof.* The energy consumption for ideal schedule is given by:

$$E_o = T_o \left( \sum_{i=1}^{r_o} \mu(c_i) + \Gamma \right) \quad (5.28)$$

Since we do not know which machines are working for time more than  $T_o$  and which ones lesser than  $T_o$ , for the upper bound we assume that all machines are working for  $T^*_{max}$ . Hence,

$$E^*_{max} = T^*_{max} \left( \sum_{i=1}^{r_o} \mu(c_i) + \Gamma \right) \quad (5.29)$$

Clearly,

$$\frac{E^*_{max}}{E_o} = \frac{T^*_{max} \left( \sum_{i=1}^{r_o} \mu(c_i) + \Gamma \right)}{T_o \left( \sum_{i=1}^{r_o} \mu(c_i) + \Gamma \right)} \quad (5.30)$$

$$\frac{E^*_{max}}{E_o} = \frac{T^*_{max}}{T_o} \quad (5.31)$$

By (Dobson, 1984, Theorem 4.1),

$$\frac{T^*_{max}}{T_o} \leq \frac{19}{12} \quad (5.32)$$

From (5.31) and (5.32), we get,

$$\frac{E^*_{max}}{E_o} \leq \frac{19}{12} \quad (5.33)$$

□

As the bound on energy depends on makespan in our algorithm, the worst case bound for our algorithm is also  $\frac{19}{12}$ .

## 6 Energy Efficiency: Incompatible Measures

There can be two measures for assessing the energy efficiency of a system of machines: one, to consider the total energy consumed by the system to complete some work; and the second, to consider the fraction of the energy consumed by machines in the system to do work over the total energy consumed in the system.

The first measure gives a sense of the energy required by the system per unit work produced, and the second measure indicates how much of the energy consumed by the system goes into work, and how much is idle (non-working) consumption by the system. We hold that the first measure, energy per unit work, is the more meaningful one (as it can lead to lowered overall energy consumption while completing some amount of work).

Our analysis clearly indicates that these two measures of energy efficiency are incompatible, in the sense that they cannot be simultaneously optimized for arbitrary systems. To see why, we may consider (4.43) for the total energy of the system. In (4.43), if we put  $\gamma(c_i) = 0, \forall i, 1 \leq i \leq m$ , then the working energy is given by:

$$\sum_{i=1}^r \mu(c_i) \left[ \frac{W}{\sum_{i=1}^r v(c_i)} \right] \quad (6.1)$$

Dividing the above by (4.43), we get the following for the ratio of the working energy to total energy:

$$\frac{\sum_{i=1}^r \mu(c_i)}{\sum_{i=1}^r (\mu(c_i) - \gamma(c_i)) + \Gamma} \quad (6.2)$$

This in turn simplifies to:

$$\frac{\sum_{i=1}^r \mu(c_i)}{\sum_{i=1}^r \mu(c_i) + \sum_{i=r+1}^m \gamma(c_i)} \quad (6.3)$$

Considering (6.3) shows that the ratio of the working energy to total energy can only be optimized by increasing the value of  $r$ , i.e., when  $r = m$ , but this violates Theorem 4.4 which tells us that for minimum total energy consumption, the value of  $r$  may not be necessarily be equal to  $m$ .

Thus, a system running in such a way as to consume the least possible energy while completing some work will not always be most efficient in terms of the second measure, as the energy consumption by the idle machines can be significant. Conversely, if we seek to optimize the system performance by the second measure and reduce the idle consumption of machines, then the overall energy consumption to do the work is not always minimized.

The data center energy efficiency metric called Power Usage Effectiveness (PUE) (Avelar et al., 2012), an industry standard recommended by the U.S. Environmental Protection Agency under its Energy Star program,<sup>1</sup> is an energy efficiency measure of the second kind, as it considers only the ratio of the total energy to the working energy. It is thus not a surprise that the PUE comes with its share of controversy and criticisms (Brady et al., 2013).

---

<sup>1</sup>See [https://www.energystar.gov/ia/partners/prod\\_development/downloads/DataCenterRating-General.pdf](https://www.energystar.gov/ia/partners/prod_development/downloads/DataCenterRating-General.pdf).

## References

- Abdullah, M. and M. Othman  
2013. Cost-based multi-QoS job scheduling using divisible load theory in cloud computing. *Procedia Computer Science*, 18:928–935.
- Agrawal, P. and S. Rao  
2014. Energy-aware scheduling of distributed systems. *IEEE Trans. Autom. Sci. Eng.*, 11(4):1163–1175.
- Agrawal, P. and S. Rao  
2015. Energy-minimal scheduling of divisible loads. In *4th International Workshop on Energy-Efficient Data Centres (E2DC 2015), co-located with ACM E-Energy 2015*.
- Albers, S.  
2009. Algorithms for energy saving. In *Efficient Algorithms*, Pp. 173–186. Springer.
- Augustine, J., S. Irani, and C. Swamy  
2004. Optimal power-down strategies. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, Pp. 530–539. IEEE.
- Avelar, V., D. Azevedo, and A. French  
2012. *PUE: A Comprehensive Examination of the Metric*. The Green Grid. White Paper #49.
- Bambagini, M., M. Bertogna, M. Marinoni, and G. Buttazzo  
2013. An energy-aware algorithm exploiting limited preemptive scheduling under fixed priorities. In *8th IEEE International Symposium on Industrial Embedded Systems (SIES 2013)*, Pp. 3–12.
- Bansal, N., H.-L. Chan, and K. Pruhs  
2009. Speed scaling with an arbitrary power function. In *Proceedings of the twentieth annual ACM-SIAM symposium on discrete algorithms*, Pp. 693–701. Society for Industrial and Applied Mathematics.
- Bansal, N., H.-L. Chan, and K. Pruhs  
2013. Speed scaling with an arbitrary power function. *ACM Transactions on Algorithms (TALG)*, 9(2):18.
- Bansal, N., T. Kimbrel, and K. Pruhs  
2007. Speed scaling to manage energy and temperature. *Journal of the ACM (JACM)*, 54(1):3.
- Berral, J. L., I. Goiri, R. Nou, F. Juliá, J. Guitart, R. Gavalda, and J. Torres  
2010. Towards energy-aware scheduling in data centers using machine learning. In

- e-Energy '10: Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, Pp. 215–224.
- Brady, G. A., N. Kapur, J. L. Summers, and H. M. Thompson  
2013. A case study and critical assessment in calculating power usage effectiveness for a data centre. *Energy Conversion and Management*, 76:155–161.
- Dobson, G.  
1984. Scheduling independent tasks on uniform processors. *SIAM Journal on Computing*, 13(4):705–716. doi:10.1137/0213044.
- Drozdowski, M., J. Marszałkowski, and J. Marszałkowski  
2014. Energy trade-offs analysis using equal-energy maps. *Future Generation Computer Systems*, 36:311–321.
- Graham, R. L.  
1969. Bounds on multiprocessing timing anomalies. *SIAM journal on Applied Mathematics*, 17(2):416–429.
- Halliday, D., R. Resnick, and J. Walker  
2010. *Fundamentals of physics extended*, volume 1. John Wiley & Sons.
- Herrmann, J. W., ed.  
2006. *Handbook of Production Scheduling*, volume 89 of (*International Series in Operations Research & Management Science*). Springer.
- Hirofuchi, T., H. Nakada, H. Ogawa, S. Itoh, and S. Sekiguchi  
2010. Eliminating datacenter idle power with dynamic and intelligent VM relocation. In *Distributed Computing and Artificial Intelligence*, volume 79 of *Advances in Intelligent and Soft Computing*, Pp. 645–648. Springer Berlin Heidelberg.
- Huang, J., C. Buckl, A. Raabe, and A. Knoll  
2011. Energy-aware task allocation for network-on-chip based heterogeneous multiprocessor systems. In *19th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP 2011)*, Pp. 447–454, Cyprus.
- Irani, S. and K. R. Pruhs  
2005. Algorithmic problems in power management. *ACM Sigact News*, 36(2):63–76.
- Johnson, D. S.  
1973. Approximation algorithms for combinatorial problems. In *Proceedings of the fifth annual ACM symposium on Theory of computing*, Pp. 38–49. ACM.
- Le Sueur, E. and G. Heiser  
2010. Dynamic voltage and frequency scaling: The laws of diminishing returns. In

*HotPower '10: Proceedings of the 2010 International Conference on Power Aware Computing and Systems*, Vancouver, Canada.

Lee, K., J. Y. Leung, Z.-h. Jia, W. Li, M. L. Pinedo, and B. M. Lin  
2014. Fast approximation algorithms for bi-criteria scheduling with machine assignment costs. *European Journal of Operational Research*, 238(1):54–64.

Lee, K., J. Y.-T. Leung, and M. L. Pinedo  
2012. Coordination mechanisms for parallel machine scheduling. *European Journal of Operational Research*, 220(2):305–313.

Leung, J. Y.-T., K. Lee, and M. L. Pinedo  
2012. Bi-criteria scheduling with machine assignment costs. *International Journal of Production Economics*, 139(1):321–329.

Pierson, J.-M., ed.  
2015. *Large-Scale Distributed Systems and Energy Efficiency*. John Wiley and Sons. Wiley Series on Parallel and Distributed Computing.

Pinedo, M. L.  
2009. *Planning and Scheduling in Manufacturing and Services*, 2 edition. Springer.

Pinedo, M. L.  
2012. *Scheduling: theory, algorithms, and systems*. Springer.

Pouwelse, J., K. Langendoen, and H. Sips  
2001. Energy priority scheduling for variable voltage processors. In *In Intl. Symp. on Low-Power Electronics and Design*, Pp. 28–33.

Robertazzi, T. G., M. A. Moges, and D. Yu  
2005. Divisible load scheduling with multiple sources: Closed form solutions. In *Conference on Information Sciences and Systems*.

Sheikh, H. F., H. Tan, I. Ahmad, S. Ranka, and P. B. V.  
2012. Energy- and performance-aware scheduling of tasks on parallel and distributed systems. *J. Emerg. Technol. Comput. Syst.*, 8(4):32:1–32:37.

Shi, H., W. Wang, and N. Kwok  
2012. Energy dependent divisible load theory for wireless sensor network workload allocation. *Mathematical Problems in Engineering*.

Singh, A. K. and S. Sahu  
2014. Environment conscious public cloud scheduling algorithm with load balancing. *International Journal of Computer Applications*, 87(13):24–27.

- Veeravalli, B., D. Ghose, and T. G. Robertazzi  
 2003. Divisible load theory: A new paradigm for load scheduling in distributed systems. *Cluster Computing*, 6(1):7–17.
- Wang, Y. and M. Saksena  
 1999. Scheduling fixed-priority tasks with preemption threshold. In *Sixth International Conference on Real-Time Computing Systems and Applications (RTCSA '99)*, Pp. 328–335, Hong Kong, China.
- Yao, F., A. Demers, and S. Shenker  
 1995. A scheduling model for reduced cpu energy. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, Pp. 374–382. IEEE.
- Yassa, S., R. Chelouah, H. Kadima, and B. Granado  
 2013. Multi-objective approach for energy-aware workflow scheduling in cloud computing environments. *The Scientific World Journal*.
- Yu, D. and T. G. Robertazzi  
 2003. Divisible load scheduling for grid computing. In *Fifteenth IASTED International Conference on Parallel and Distributed Computing and Systems*, volume 1, Pp. 1–6.
- Zomaya, A. Y. and Y. C. Lee, eds.  
 2012. *Energy-Efficient Distributed Computing Systems*. Wiley-IEEE Computer Society.